DOCUMENT RESUME

ED 128 177 SE 020 952

AUTHOR Goldenberg, E. Paul
TITLE A Glossary of PDP11 LOGO Primitives. Artificial Intelligence Memo Number 315A.
INSTITUTION Massachusetts Inst. of Tech., Cambridge. Artificial Intelligence Lab.
SPONS AGENCY National Science Foundation, Washington, D.C.
REPORT NO LOGO-16
PUB DATE Mar 75
GRANT NSF-GJ-1049
NOTE 40p.; Adapted from ED118370
AVAILABLE FROM The Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139 ($1.30)

EDRS PRICE MF-$0.83 HC-$2.06 Plus Postage.
DESCRIPTORS Artificial Intelligence; *Computer Programs; Computers; *Computer Science Education; Glossaries; *Manuals; *Mathematics Education; *Programming Languages

ABSTRACT

This manual provides brief descriptions of the primitives and is the PDP11 implementation of the LOGO language. It is intended as a quick reference for users familiar with the LOGO language. (SD)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED128177

ARTIFICIAL INTELLIGENCE
MEMO 315A

LOGO
MEMO 16

A GLOSSARY OF PDP11 LOGO PRIMITIVES

March 18, 1975

E. Paul Goldenberg

Adapted from Memo 14 of Abelson and Adams

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

This glossary was written for the purpose of providing a quick and concise yet accurate description of the primitives and special words and characters of the March 18, 1975 PDP 11 implementation of the LOGO language. Many entries include references to other related words and/or examples of the use of the primitive being described, but this is not intended to replace the function of a good manual. For a more detailed and comprehensive description of the language, see the LOGO MANUAL, LOGO MEMO 7.

The description of each LOGO word includes the word, itself, any arguments that the word may require, the "type" of word it is, abbreviated and alternate forms of the word, if any, and a definition correct as of the date of this glossary. Word type is described on the first page and an example of the format of the entries is given below.

In the appendix to this glossary are sections about 1) LOGO words that take a VARIABLE NUMBER OF INPUTS, 2) INFIX OPERATORS, 3) EDITING CHARACTERS, 4) SPECIAL CHARACTERS, 5) SPECIAL NAMES, 6) DECIMAL ASCII CODE AND CORRESPONDING CHARACTERS.

PRIMITIVE *inputs (optionalinputs)* (word type)
Brief description and references to other LOGO PRIMITIVES.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's education research is provided in part by the National Science Foundation under grant #GJ1049.

SE 020 952

WORD TYPE

Some LOGO words output values to other LOGO words. We call these words "Operations." LOGO words which do not output are generally called "Commands," but some further subdivision is used in this glossary. Words whose purpose is to ***COMMAND*** a device to do something now (whose purpose is not, primarily, to switch into some mode that affects the behavior of future commands or to monitor the operation of a procedure) are called COMMANDS. Words whose primary purpose is to ***SWITCH*** to another mode or device (e.g., making FORWARD apply to the TouchSensor-Turtle instead of the display, or entering DEBUG mode) are called SWITCHES. Words that determine the ***FLOW*** of control in a procedure (e.g., STOPping, PAUSEing, WAITing, GOing to new lines, etc.) are called FLOW words. LOGO words that cannot be used without specially unlocking their protection are called STATUS words. There is also a series of words that serve only to clarify the nature of the word following them. These words are called MODIFIERS in this glossary. To help the reader, the COMMANDS, OPERATIONS, SWITCHES, and FLOW words, are further modified, where appropriate, with labels indicating which specific device or devices they apply to. Words that apply generally are not further labeled.

ALL (modifier)

Used in expressions like PRINTOUT ALL or ERASE ALL to indicate that the command applies to your entire workspace. In addition to the two commands already mentioned, ALL may sensibly follow TRACE, STEP, BURY, ERASE TRACE, ERASE STEP, and ERASE BURY.

ALSO *anothercontrolcommand* (switch)

After you have elected to control one device, such as a floorturtle, display or plotter, you may reserve another for your control with this command. (Actually ALSO doesn't care if you are already controlling anything or not.) To use a display and a turtle, for example, say:

```
?CLEARSCREEN
?ALSO TURTLE 1
?
```

Then ambiguities about which device you are directing commands to are cleared up by using the CNTRL command. The (unquoted) inputs that ALSO understands are TURTLE 1, TURTLE 2, PLOTTER, and DISPLAY 8, DISPLAY 1 (see STARTDISPLAY for explanation of the inputs to DISPLAY.)

ARCTAN *number* (operation)
ATAN *number*

Outputs the arctangent of *number* in degrees.

ARRAY *arrayname* (modifier)

Indicates that *arrayname* is the name of an array that has already been defined by DEFAR. Used in combinations such as ERASE ARRAY "CALENDAR.

ARRAYS (modifier)

Plural of ARRAY. ER ARRAYS or PO ARRAYS, applies to all arrays in your workspace.

ASIZE *arrayname* (operation)

Outputs the dimensions of the array *arrayname* as a list.

ATOD *devicenumber* (operation)

Analog to Digital. Outputs the current value of device number *devicenumber*. E.g., PR ATOD 6 prints a number corresponding to the horizontal position of one of the joysticks. (Time value information: true as of Sept. 7, 1974, evening.)

BACK *distance* (turtle command)
BK *distance*

Causes the display, plotter or floor turtle to back up *distance*.

BELL (command)

Rings a bell at your console (if your console has a bell), as does .CTYO 7

BITOUT *number* (command)

This outputs 16 bits to 16 relays at once, by converting *number* into binary and sending the lowest order bit to relay 1, the next to relay 2 and so on. High order zeros are high order zeros.

BOTH *TorF1 TorF2* (operation)

Outputs the logical and of both its inputs which must be either "TRUE or "FALSE. That is, it outputs "TRUE if and only if both of its inputs are (or evaluate to) "TRUE.

BOXIN *switchboxnum* (operation)

Outputs a number which is the sum of the powers of 2 corresponding to switches that are on at the time this operation is executed. (Clear?) For example, BOXIN 3 outputs 34 if switches 5 and 1 ($2^5 + 2^1$) on box 3 are on, and 0 if no switches are on. See appendix for a diagram of the switchbox switch numbers.

BTOUCH (TS-turtle operation)

Outputs "TRUE if the back of the touch-sensor turtle is touching something. Outputs "FALSE otherwise. Like FTOUCH, RTOUCH, and LTOUCH.

BURY *someprocedure* (command)

"Buries" the procedure *someprocedure*. A buried procedure may be executed like any other, may be printed out if specified by name and may be edited (though editing of the title unBURIes it even if no title change is made by the edit). But the procedure is "invisible" to CONTENTS, PO ALL, ERASE ALL and WRITE. Thus, one convenient way of saving a procedure while liberating yourself from many others is to BURY the good one, ERASE ALL and then ER BURY the good one. You can now write that one into a file. Only procedures can be BURIed. BURY quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix. Like other commands that quote their arguments, no quote is needed on *someprocedure*, but the presence of a " does not change the effect (enabling you to execute computed BURIes).

BUTFIRST *wordorlist* (operation)
BF *wordorlist*

If *wordorlist* is a list, BF outputs the list that contains all but the first element of the list *wordorlist*.

```
?PRINT BF [DO YOU LOVE ME]
YOU LOVE ME
?
```

If *wordorlist* is a word, BF outputs the word that is *wordorlist* with its head cut off.

```
?PR BF "WHERE
HERE
?
```

If *wordorlist* is the empty list or the empty word, BF can't chop off its leading edge and complains. See also FIRST, LAST and BUTLAST.

BUTLAST *wordorlist* (operation)
BL *wordorlist*

The same as BUTFIRST, but last.

.CASESW *0or1* (switch)

The input 0 turns off the feature that converts lower case alphabetic characters typed at LOGO to upper case. Normal mode is restored by typing .CASESW 1.

.CGCF (status switch)

This undoes the effect of .SGCF

CHAR *asciivalue* (operation)

Outputs the character whose decimal ascii number is *asciivalue*. *See appendix.*

CLEARSCREEN (command and switch)
CS

Erases all lines from a display and places the turtle in the center of the display facing up. CS does not alter the turtle's visibility or its readiness to draw. (That is, if the turtle was hiding, or if its pen was up, those states are not reset as they are by STARTDISPLAY.)

CLOCK (operation)

This outputs a number which is incremented every 60th of a second. One can thus use CLOCK to time things.

.CLOSE *device* (command)

Closes output to the device you are controlling and frees it for someone else to use it. Its input may either be the number of a device or a quoted device name. The device names it recognizes are "LIGHT, "TUR1, "TUR2, "MUSIC, "PLOTTER and "TTY which you can look up in the section on LOGO SPECIAL NAMES in the appendix. See .TYO also.

.CLOSEF (file command)

Closes whatever file is currently open (for reading or writing). This must be executed after .OPENR, .OPENW, and .OPENA before any other file opening commands (the three previous or READ, WRITE, POI, POF, etc.) can be executed. See FILE for details.

CNTRL *controlcommand* (switch)

Directs following turtle, display or plotter commands to appropriate device until another CNTRL or control command. Used with ALSO. It's (unquoted) input must be TURTLE 1, TURTLE 2, PLOTTER, or DISPLAY.

CONTENTS (operation)

Outputs a list of titles of all (unBURIed) procedures in the user's workspace.

CONTINUE (flow)
CO

Continues the execution of a procedure that has been PAUSEd by PAUSE, ^Z, or an error while in DEBUG mode. Execution starts at the next line after the PAUSE and not at the PAUSE itself.

COS *degrees* (operation)

Outputs the cosine of *degrees*.

COUNT *thing* (operation)

If the input is a list, COUNT outputs the number of members of the list. If the input is a word, this outputs the number of letters in the word.

```
?PR COUNT [HEY' OLD ARE YOU TODAY?]
5
?PR COUNT "YOURFINGERS
11
?
```

.CPNF (status switch)

This undoes the effect of .SPNF

CRINDEX *indexname* (file command)

This command creates an index on whatever branch of the filing tree you currently happen to be hacking at. See FILE for more details.

.CTF (status switch)

This undoes the effect of .STF

.CTYI (operation)

This takes no input but demands a character from the console when executed and outputs that character's decimal ascii value. Be warned that .CTYI treats all characters alike without showing partiality to ^G's or the BREAK key. That is, if you do not provide a test for these keys in your program, they will not help you to stop your procedure.

.CTYO *ascii* (command)

This outputs a character to your console. The console then handles that character in whatever way it's instincts tell it. In particular, .CTYO 65 causes any console to type an A whereas .CTYO 29 and .CTYO 31 both do nothing at all on the printing consoles, but are Homeup and Erase to EOF respectively on the Datapoint.

CTYOWAIT *extrawait* (flow)

Causes your program to wait until your console has finished typing out before continuing with the next instruction. This is useful for proper synchronization of TTY output and display, musicbox or other devices. The input specifies an additional wait after the completion of typing.

DATE (operation)

Outputs a list which might contain the current date, but probably will not. If the date is once set correctly (see .SETTV), then DATE will, in fact, output the correct thing, even if the system remains up for a whole day. The elements of the list are [yr mo da].

DEBUG (switch)

Turns debug mode on and off. When in debug mode, errors cause a PAUSE rather than a BREAK to the top level. During a pause (reached this way or by using ^Z or PAUSE) the user may execute other instructions (e.g., PO NAMES to find out the state of his variables including local variables if one is not at level 0) and then continue execution by using the CO command.

DEFINEARRAY *aname dim type* (command)
DEFAR *arrayname dimensions type*

Arrays may be one-, two- or three-dimensional but parentheses must be used around DEFAR and its inputs if a more-than-one dimensional array is to be created. For example:

```
?( DEFAR "SOMEARRAY  :DIM1 :DIM2 :DIM3 2 )
?
```

creates a three-dimensional "pointer" array called "SOMEARRAY. The type must be 0 for integer entries into the array, 1 for floating point entries, and 2 for lists and words (and numbers as well, though these are treated as words when they get got by GET). All cells of a newly created type 0 or 1 array contain 0 until you STORE something else in them. Pointer arrays start out with empty cells. Indexing of arrays begins at 0, so that referencing of :DIM1 runs from 0 through :DIM1 - 1.

DELETE *filename* (file command)

See ERF (ERASE FILE).

DELETEINDEX *indexname* (file command)

See ERI (ERASE INDEX).

.DEPOSIT *coreaddress* (status command)

Don't play with this without knowing much about the guts of LOGO! It can wipe out lots of good things. It requests a numerical input from the console and deposits it directly into *coreaddress*.

DIFFERENCE *subtrahend minuend* (operation)

Wow, it's been a long time since I've had a chance to use those words!! See section on INFIX OPERATORS in the appendix.

DISPLAY *snapname* (display command)

Displays a snap (see SNAP) in the same location relative to the current position of the turtle that the original drawing was to the position of the turtle at the time the snap was made. (I know that was hard.) The turtle's position after the snap is displayed will be the position the turtle had on the snap when the snap was first made. At present, the displayed snap always appears in the orientation in which it was originally drawn regardless of the heading of the turtle. DISPLAY is also used (without an input) as an input to CNTRL and ALSO.

.ECHOSW *0or1* (switch)

The input 0 turns off the feature that echos characters typed at the console. Characters typed will be heard by LOGO, but will not appear to have been typed (the console looks dead unless other characters are being outputted to the console). Except for characters being typed in, no printing is suppressed (i.e., PRINT, .CTYO, etc. will still act normally). .ECHOSW 1 restores normal mode.

EDIT *someprocedure* (switch)
ED *someproc*

The procedure named is designated for editing and editing mode is entered. The prompt character for editing mode, as for defining a procedure is the > rather than the ? that indicates execution mode. To switch off editing mode (when done editing the procedure or to avoid unwanted side-effects) type END. EDIT quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.

EDL *linenumber* (editing command)
EDIT LINE *linenumber*

Places line number *linenumber* of the procedure that is being edited in the edit buffer where it can be manipulated using the special editing control characters (see EDITING CHARACTERS in appendix).

EDT (editing command)

Enables the user to edit the title of the procedure he is defining or editing. He may either use the special editing control characters or simply retype the title. Using this command is the only way of editing the title of a procedure.

EITHER *TorF1 TorF2* (operation)

Outputs "TRUE if EITHER of the inputs are "TRUE, "FALSE if both are "FALSE. The logical or. Both inputs must be or evaluate to "TRUE or "FALSE.

ELSE *otherinstructions* (control word)

Can be used with IF (and THEN) to allow an alternative course of action to take place if the conditional in an IF-THEN pair is "FALSE.

```
?MAKE "WEATHER "CLOUDY
?IF :WEATHER = "SUNNY PRINT "GOOD ELSE PRINT "TOUGH
TOUGH
?
```

EMPTYP *wordorlist* (operation)

Outputs "TRUE if the argument is the empty word or the empty list, "FALSE otherwise.

END (switch)

Tells LOGO you are finished editing or defining a procedure. Switches off editing mode.

EQUAL *this that* (operation)

Outputs "TRUE if *this* and *that* are equal or evaluate to the same thing.

```
?PRINT EQUAL 2    1.5 * 2 - 1
TRUE
?IF EQUAL "BOY "GIRL PR "HMM! ELSE BELL
?                 ! bell rings at your console
```

See section on INFIX OPERATORS in appendix.

ERASE *procedurenameormodifier* (command)
ER *procedurenameormodifier*

Erases the procedure from workspace. A procedure may not be erased while it is being edited or defined. ERASE quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix. ERASE can be followed by a modifier instead of a procedure name, in which case it acts accordingly. In all cases, either ERASE or ER may be used. Three cases (ERF, ERI, and ERL) may be abbreviated even further.

ER ALL (command)

Erases everything in workspace except for BURIed procedures. This command clears the stack, that is, stops the execution of any following instructions on the same line or in the same procedure, even if the procedure is buried. It cannot be executed in edit mode.

ER ARRAY *arrayname* (command)

ER ARRAYS (command)

Erases all arrays.

ER BURY *procedurename* (command)

Unburies the named procedure. If *procedurename* is ALL, this unburies all procedures. No error is produced if you try to unbury a procedure that exists but is not buried. ER BURY quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.

ER FILE *filename* (file command)
ERF *filename*
DELETE *filename*

Erases the file from the disk. This does not affect your workspace.

ER INDEX *indexname* (file command)
ERI *indexname*
DELETEINDEX *indexname*

Deletes the index *indexname* from the branch of the filing tree you are sitting on if that index contains no files. See FILE for more details.

ERL *linenumber* (editing command)
ER LINE *linenumber*

Removes that line from the procedure you are in.

ER NAME *name* (command)

Removes the global variable *name* from the workspace.

ER NAMES (command)

Erases all global variables ("names") from the workspace.

ER PROCEDURES (command)

Erases all unburied procedures from the workspace, but leaves variables and arrays untouched.

ER STEP *procedurename* (switch)

UnSTEPs the procedure (but does not erase anything else.) If ALL is the *procedurename*, this unSTEPs all STEPped procedures. You cannot unSTEP a STEPped procedure while it is running (or during a PAUSE in its running) but no error is encountered if you try. The ER STEP takes effect after the procedure stops. ER STEP quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.

ER TRACE *procedurename* (switch)

UnTRACEs the procedure. As with ER STEP, ER TRACE ALL works, but also as above a procedure cannot be unTRACEd while it is running. ER TRACE quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.

ERBRK (operation)

If used in an ERSET procedure, ERBRK outputs 1 if the running was interrupted by pressing ^G, -1 if it was interrupted by pressing ^Z, 0 if it was interrupted by an error condition.

ERCLR (switch)

Error clear. Undoes the effect of ERSET.

ERLIN (operation)

Outputs the procedure line number in which the last error occurred. This outputs 0 if the last error was a top level error or if it occurred just as a procedure was being invoked (the title line is considered line 0).

ERLOC (operation)

Outputs the location in the computer's core at which the last error occurred.

ERNAM (operation)

Outputs a fairly reasonable abbreviated name of the last error. Examples: "UBL = used by logo, "FNF = file not found, "HNM = has no meaning. The best way to find out the name of a particular error is to generate the error and then look at the ERNAM.

```
?TO LEARNERNAMS
>10 OUTPUT ERNAM
>END
LEARNERNAMS DEFINED
?ERSET LEARNERNAMS
?4
WTO
?NOTHING
HNM
?
```

ERNUM (operation)

Outputs the number of the last error. (But you don't really want to know that, do you?)

ERPRO (operation)

Outputs the name of the procedure in which the last error occurred. ERPRO outputs the empty word if the last error was at top level.

ERRET *linenumber* (command)

When used in an ERSET procedure ERRET returns execution to line *linenumber* of the procedure in which the last error occurred.

ERSET *procedurename* (switch)

Causes the named procedure to be executed whenever an error (or break) occurs. If the procedure does not output (see OUTPUT), LOGO saves up the error message until all currently running procedures have stopped and then prints the message (the last message if more than one error is encountered). If the ERSET procedure does output, the output is printed instead of the normal error message. System bug errors and the error "NSL (no storage left) cannot be overridden by ERSET. If an error is encountered *within* the procedure that has been ERSET, ERCLR is executed.

ERTOK (operation)

Outputs the number of the token in the line in which the previous error occurred. If you are versed in the mysteries of the LOGO evaluator this might be useful to you.

.EXAMINE *coreaddress* (status command)

Prints the contents (octal) of *coreaddress* in the format, *coreadress/contents*.

FILE *filename* (modifier)

Used in expressions like PO FILE "MAIL or ER FILE "GARBAGE to indicate that the command applies to a file and not to something in your workspace.

GENERAL INFORMATION ABOUT THE FILING SYSTEM.

All files must be closed before any other file-referencing comands can be executed. The file referencing commands are: .OPENA, .OPENR, .OPENW, CRINDEX, DELETE, DELETEINDEX, ER FILE, ER INDEX, ERF, ERI, LOGIN, MAIL, PO FILE, PO INDEX, PO TREE, POF, POI, READ, SETINDEX, and USE. Of these, only the first three leave files open (to be closed by .CLOSEF.) It is worth remembering that when executing direct commands from a file, none of these commands can reference another file while the file they are being executed from remains open. .FILER closes an *n* line file when it tries to read line *n+1*. Inputs for all file-referencing commands that take inputs (except LOGIN) can either be words, in which case they refer to the subordinate file or index, or lists, in which case all but USE and MAIL interpret the list as a tree beginning with the next subordinate index (or with the root if the first list element is FIXED;, AI01;, or just ;). USE and MAIL begin with a user whose name must be the first element of the list. See TREE for more information about the structure of the filing system.

.FILEO (operation)

Outputs "TRUE if a file is currently open and "FALSE if no file is open.

.FILEP *list* (file command)

Writes its argument in a file opened by .OPENW or .OPENA commands. Like FPRINTing into a file. If no file is open for writing, .FILEP causes an error. See REQUEST and TEXT for examples of use.

.FILER (file operation)

Drags one line out of a file that has been opened by .OPENR (beginning at the beginning and working sequentially through the file in each successive use while the file is open) and outputs that line as a *list*. (Like REQUEST, but from a file.) When no file is opened for reading, .FILER causes an error.

FIRST *wordorlist* (operation)
F *wordorlist*

If the input is a list, FIRST outputs the first element of the list, which may be a word or a list itself. (Unlike BUTFIRST which must output a list if its input was a non-empty list.) If the input is a word (or number), FIRST outputs the first character of the word. See also LAST, BUTFIRST and BUTLAST.

```
?PRINT FIRST 637
6
?PRINT FIRST "ALPHABET
A
?FPRINT F BF [[ONE 1] [TWO 2] [THREE 3]]
[TWO 2]
?
```

FORWARD *distance* (turtle command)
FD *distance*

Tells the floor or display turtle to move ahead *distance*.

FPRINT *wordorlist* (command)

Like PRINT except that it also prints the outside brackets of a list.

```
?FPRINT [THIS LOOKS LIKE A LIST]
[THIS LOOKS LIKE A LIST]
?PRINT [BUT NOT THIS]
BUT NOT THIS
?
```

continued next page

continued from last page FPRINT can also take a variable number of inputs in which case it FPRINTs with no spaces between inputs.

```
?( FPRINT "A "B [C] [D] [E [F G]] )
AB[C] [D] [E [F G]]
?( PRINT "A "B [C] [D] [E [F G]] )
ABCDE [F G]
?
```

See PRINT also.

FPUT *wordorlist list* (operation)

Creates and outputs a new list whose FIRST is *wordorlist* and whose BUTFIRST is *list*. *List* is not destroyed. (Compare this operation with LPUT, SENTENCE and LIST.) This operation can take a variable number of inputs in which case all but the last are FPUT onto *list*.

```
?FPRINT FPUT "WORD [ONTO A LIST]
[WORD ONTO A LIST]
?FPRINT ( FPUT "WORD [AND LIST] "ANDWORD [ONTO A LIST] )
[WORD [AND LIST] ANDWORD ONTO A LIST]
?FPRINT FPUT [LIST] [ONTO LIST]
[[LIST] ONTO LIST]
```

FTOUCH (TS-turtle operation)

Outputs "TRUE if the front of the touch-sensor turtle is touching something. Outputs "FALSE otherwise. Like BTOUCH, RTOUCH, and LTOUCH.

.GCOLL (command)

Calls the garbage collector. You generally don't need this since the garbage collector is rather conscientious and works whenever he (she?) sees the need anyway.

GET *arrayname index* (operation)

Outputs contents of the cell specified by *index* in an array previously defined by DEFAR. Like DEFAR, this command requires parentheses if more than one index is needed (i.e., if the array is two- or three-dimensional).

GO *linenumber* (flow)

Used in a procedure to transfer control to that line of the procedure.

GOODBYE (command)

Prints out a sweet message and then executes HELLO.

GREATER *thisnum thatnum* (operation)

Ouputs "TRUE if *thisnum* is numerically GREATER than *thatnum*. Otherwise outputs "FALSE. It is an error if either input does not evaluate to a number. See section on INFIX OPERATORS in appendix.

.GUN *usernum* (vicious command)

Flushes the workspace and frees the console of user number *usernum*. Like having that user type GOODBYE and useful in situations where he cannot do that. Not to be used without the target's knowledge and consent.

.HALT (status command)

Stops LOGO by halting the computer.

HEADING (display operation)

Outputs the heading of the display turtle (i.e., the direction it is facing) in degrees. When the turtle is facing North (its state directly after the HOME or CS commands), its heading is 0.

HELLO (command)

Initializes the workspace of the console you are using by erasing absolutely everything from it. Whatever devices may have been in use at that console (e.g., display, music box, turtle) are released for use by others. Use HELLO to clear out unwanted garbage from the workspace you are about to use. It is also good housekeeping to say HELLO when you leave, but if this offends your sense of order, please say GOODBYE.

HERE (display operation)

Outputs a list of 3 elements, the XCOR, YCOR and HEADING of the display turtle. Exactly equivalent to (SENTENCE XCOR YCOR HEADING).

HIDETURTLE (display command)
HT

This makes the triangular display turtle invisible. It continues to behave as it did when you could see it -- in particular, it continues to leave a track. If you don't want to see the turtle track, type PENUP.

HISSPEED *devicenum speed* (switch)

Sets the computer line speed of *devicenum* to *speed*. See MYSPEED for details.

HOME (display command)

Sets the display turtle at "home," i.e. at coordinates (0,0) and heading north. This is equivalent to SETTURTLE [0 0 0].

IF *TorF logoline* (control word)

IF demands only the input *TorF* which must evaluate to "TRUE or "FALSE. If the input evaluates to "TRUE, the rest of the line (i.e., *logoline*) is executed. If *TorF* evaluates to "FALSE, the rest of the line (up to an optional ELSE) is ignored.

```
TO REACH
10 IF FTOUCH THEN TOOT 5 STOP ELSE FORWARD 10
20 GO 10
END
```

See ELSE, EQUAL and THEN for examples of the use of IF.

IFFALSE *logoline* (control word)
IFF *logoline*

Executes the *logoline* if the result of the most recent (local) TEST was "FALSE. Since results of TEST are only stored locally within a procedure, this control word always executes its line at top level.

IFTRUE *logoline* (control word)
IFT *logoline*

Executes *logoline* if the result of the previous (local) TEST was "TRUE.

ILINE (operation)

Outputs, as a list, the last line typed in at the console.

INDEX *(indexname)* (modifier)

Used in expressions such as ER INDEX "USELESSIDX or PO INDEX to indicate that it is an index and not something in your work space that you are referring to. See FILE for more details.

INTEGER *fpointnum* (operation)
INT *fpointnum*

Outputs the integer part of the floating point number *fpointnum*. The input can be in decimal or exponential form.

LAMPOFF (turtle switch)

Turns the light on the floor turtle off.

LAMPON (turtle switch)

...And the floorturtle's light turns on.

LAST *wordorlist* (operation)
L *wordorlist*

If the input is a list, LAST outputs the last element of the list, which may be a word or a list itself. If the input is a word (or number), LAST outputs the last character of the word. Analogous to FIRST. See also FIRST, BUTFIRST and BUTLAST.

LEFT *degrees* (turtle command)

Causes the floor or display turtle to rotate *degrees* to the left.

LESS *thisnum thatnum* (operation)

Outputs "TRUE if *thisnum* is numerically LESS than *thatnum*. Otherwise outputs "FALSE. It is an error if either input does not evaluate to a number. See section on INFIX OPERATORS in appendix.

LEVEL (operation)

Outputs a number which tells "how many procedures deep" the current execution is. Since levels are thought of as depth (procedures call subprocedures) smaller level numbers are higher levels. Top level is level 0, command level, when no procedure is running and instructions come directly from the user. For example,

```
?PRINT LEVEL
0
?TO WHAM
>10 PRINT LEVEL
>20 WHAM
>END
WHAM DEFINED
?WHAM
1
2
3
(etc.)
```

LIGHT (light-turtle operation)

The operation LIGHT outputs a number (0 through 63) indicating the amount of light sensed by the eye of the light-turtle. Also, the name "LIGHT is understood in the command .CLOSE "LIGHT, which releases the light-turtle.

LINE *linenum* (modifier)

Used in expressions like ERASE LINE 28, PO LINE 48, or EDIT LINE 121, to indicate that the command applies to a line of a procedure and not something in your workspace. Without the word "LINE," of course, each of the expressions above would generate an error message and an error.

LIST *thing1 thing2* (operation)

Outputs a two-element list whose elements are *thing1* and *thing2* (which may be words, numbers or lists themselves), respectively. LIST can accept one input or more than two inputs if surrounded by parentheses. For example:

```
?FPRINT ( LIST ( LIST "A [ B ] "C ) )
[ [ A [ B ] C ] ]
?
```

Also compare this operation with FPUT, LPUT and SENTENCE.

LISTP *thing* (operation)

Outputs "TRUE if *thing* is a list, and "FALSE otherwise.

LOCAL *name* (command)

In a procedure, this command causes the scope of *name* to be local to the procedure (i.e., the name is undefined outside of the procedure).

LOGIN *name* (file command)

This tells the system who you are (which information is used by the MAIL and PEEK commands), USEs *name*'s INDEX of files (if the user has already specified the correct disk), searches for an file named "INIT in that index and perfoms all commands written into the file (if it exists), and then searches for a file named "MAIL, asks the user if he wishes that file to be printed out and does so if the user types Y.

LINEPRINT *filename* (command)
LP *filename*

Causes the lineprinter to print out the named file.

LPUT *wordorlist list* (operation)

Creates and outputs a new list whose LAST is *wordorlist* and whose BUTLAST is *list*. *List* is not destroyed. (Compare this operation with FPUT, SENTENCE and LIST.) This operation can take a variable number of inputs in which case all but the last are LPUT onto *list*.

LTOUCH (TS-turtle operation)

Outputs "TRUE if the left of the touch-sensor turtle is touching something. Outputs "FALSE otherwise. Like FTOUCH, RTOUCH, and BTOUCH.

MAIL *name* (file command)

Creates (or appends to) a file called "MAIL in *name*'s INDEX of files. (See LOGIN for an explanation of that file.) The computer responds to this command by typing an arrow. Anything typed at the console at this time is regarded as mail to be sent. To end the message, the user types a line which contains only a single period (.) followed by a carriage return. If the user is logged in, MAIL will identify him to the recipient of the mail file. If the input to MAIL is a tree beginning with a username, the MAIL file will be created in the specified subindex.

MAKE *name thing* (command)

Assigns the name *name* to the thing *thing*.

```
?PRINT "MY.AGE
MY.AGE
?PRINT :MY.AGE
"MY.AGE HAS NO VALUE
?MAKE "MY.AGE 9
?PRINT "MY.AGE
MY.AGE
?PRINT :MY.AGE
9
?
```

MCLEAR (music command)

Clears the music buffer.

MLEN (music operation)

Outputs the length of the longest voice loaded in the music buffer in units. See NOTE for explanation of units.

MUCTRL *voicespecs* (switch)

Voicespecs is a two-digit number, the first digit of which specifies the voices to play from and the second of which specifies the voices to load according to the following table.

DIGIT	FIRSTDIGIT	SECONDDIGIT
0	play none	load 0, 1, 2, 3
1	play 3	load 1, 2, 3
2	play 2, 3	load 2, 3
3	play all	load 3

Voices are loaded in cyclic order and played simultaneously beginning immediately after the last expected voice is loaded.

MUTYO *pitch1 pitch2* (music command)

Actually, MUTYO takes a variable number of inputs (requiring parentheses if the number of inputs is not exactly two) and loads the pitches sequentially into the music box according to the format specified by MUCTRL. Regardless of the format specified, four successive MUTYOs with two inputs each are equivalent to eight with one input each, or one with seven inputs followed by one with one. Legal inputs are -28 through 35 inclusive, except for -25 which is a special control character. Pitches progress chromatically beginning two octaves below middle C (pitch = -24) and ending three octaves above (pitch 32). Middle C is pitch 0. The percussion sounds BOOM and SSH are, -27 and -26 respectively, and a silence is -28.

MUWAIT *extrawait* (music command)

Causes your program to wait *extrawait* 30ths of a second after music output has stopped before continuing.

MYSPEED *linespeed* (switch)

Sets the line speed of your own console to *linespeed*. .PWRCLR sets everything back to the default speed, 300 BAUD (MYSPEED 7). DATAPOINT consoles set at 300 BAUD run well at the same linespeed as the other consoles, but can be set to 2400 BAUD if MYSPEED 11 is first run.

NAME *name* (modifier)

Used in expressions like ER NAME *name* or PO NAME *name* to indicate that you are referring to a name and not a procedure.

NAMES (modifier)

Plural of NAME. PO NAMES or ER NAMES refers to all names in your workspace.

NEWSNAP (display switch)

Causes the image currently on the screen not to be part of subsequent SNAPs. Also sets the starting location of subsequent SNAPs to the current position of the turtle rather than to (0,0). See also DISPLAY and SNAP.

.NODES (operation)

Outputs the number of free nodes in your workspace. If you want to know how many nodes you REALLY have, call the garbage collector first and type only one line, for example:

```
?.GCOLL PR .NODES
2286
?
```

NODISPLAY (switch)

Turns off and releases the display.

NOMUSIC (switch)

Turns off and releases the music box.

NOPLOTTER (switch)

Turns off and releases the plotter.

NOT *TorF* (operation)

Outputs "TRUE if the input *TorF* is (or evaluates to) "FALSE and outputs "FALSE if the input is "TRUE.

NOTBOX (switch)

Parsed "NO TBOX," this declares you are not a TBOX. See TBOX.

NOTE *pitch duration* (music command)

Loads one note of music into the music buffer (which may later be played by using the command PM). Pitches are numbered chromatically from -24 to 35 with 0 being middle C. There are also three special "pitches":

-28 is a silence ("rest")
-27 is the percussion sound "boom"
-26 is the percussion sound "ssh"
-25 is not a valid pitch

Duration must be between 0 and 127 units. Each unit is 1/8 second long if NVOICES 4 was used. (See NVOICES.) If *duration* is 0, NOTE generates nothing. If it is 1, the command generates a note 1 unit long. If *duration* is greater than 1, the note generated is *duration* - *1* units long folowed by 1 unit of rest (so that the music will not sound "slurred"). If *pitch* is -26 or -27, NOTE generates a 1 unit percussion sound followed by *duration* - *1* units of silence.

NOTURTLE (switch)

Turns off and releases any floor turtles that you are controlling.

NOWRAP (display switch)

Tells the computer not to allow the display turtle to go out of bounds. See WRAP.

NUMBERP *input* (operation)

Outputs "TRUE if *input* is a number and "FALSE otherwise.

NVOICES *number* (music switch)

Specifies the number of voices to which music will later be sent (using VOICE to indicate "which voice now" and NOTE to generate the music). When outputting from the music buffer to the music box, the music system normally multiplexes among four voices, but it is possible, using NVOICES to use only one or two voices. NVOICES 1 outputs only to voice 1 (and, when played, plays only from voice 1). NVOICES 2 outputs to voices 1 and 2. NVOICES 4 is the normal mode. Since the music box is fed at a constant rate, NVOICES 1 causes the basic unit of duration to be 1/4 as long as normal and NVOICES 2 causes the duration to be 1/2 as long. NVOICES 3 is not a valid state.

.OPENA *filename* (file command)

Opens a file for appending with .FILEP -- whatever was in the file remains and you add new stuff at the end. (If no file by that name existed, one is created.) Close file with .CLOSEF before leaving your console or HELLOing as files left open for writing are flushed when your workspace goes away. See FILE for more details.

.OPENR *filename* (file command)

Opens a file for reading with .FILER operation. See FILE for more details.

.OPENW *filename* (file command)

Opens a file for writing with .FILEP -- if the file already exists you will be notified and asked if you wish the contents to be deleted as with the command WRITE. See FILE for more details.

OUTPUT *whatever* (flow operation)
OP *whatever*

Used to make procedures operations instead of commands, OUTPUT tells the procedure to stop and output *whatever*.

```
?TO DOUBLE :X
>10 OP WORD :X :X
>END
DOUBLE DEFINED
?PRINT DOUBLE DOUBLE "HA
HAHAHAHA
?
```

A use in recursive procedures. Notice line 30 especially.

```
?TO MEMBERP :ITEM :LIST
>10 IF EMPTYP :LIST OP "FALSE
>20 IF :ITEM = F :LIST OP "TRUE
>30 OP MEMBERP :ITEM BF :LIST
>END
MEMBERP DEFINED
?
```

PAUSE (command)

When executed in a procedure, it causes a pause in the procedure during which the user may execute any commands he wishes. To resume the execution of the procedure (beginning on the next line after the PAUSE regardless of the location of that command on the previous line) the user uses the CONTINUE command.

PEEK (command)

Prints system usage information (who's logged in and what devices are used).

PEEKD (command)

Prints device usage information only.

PENDOWN (turtle command)
PD

Allows the floor or display turtle to draw a line when it moves next.

PENP (turtle operation)

Outputs "TRUE if the pen is down (ready to trace) and "FALSE otherwise.

PENUP (turtle command)
PU

Causes the floor or display turtle not to draw a line when it moves next.

PLOTTER (plotter switch)

Executed as a command, PLOTTER causes subsequent appropriate turtle commands to be sent to the plotter for execution. (Display commands which cannot be executed by the plotter, such as CLEARSCREEN, WRAP, DISPLAY, etc., will be ignored and will not cause an error.) Plotter argument range is more restricted than the display range. Also, the name "PLOTTER is understood in the command .CLOSE "PLOTTER, which releases the plotter for use by others. See also NOPLOTTER.

PM (music command)

Causes the output of previous NOTE commands to be played on the music box. This also clears the music buffer so that music must be recompiled before being played again.

PRINT *wordorlist* (command)

PRINTs its input followed by a carriage return. If the input is (or evaluates to) a list, PRINT does not show the outside brackets. See FPRINT for examples of PRINTing of lists and use of more than one input.

PRINTOUT *(proc.nameormodifier)* (command)
PO *(procedurenameormodifier)*

Prints the text of the named procedure on the console. If no input is specified, the default is the procedure most recently (or currently being) defined or edited including procedures defined by reading in from a file. PRINTOUT quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix. PRINTOUT can be followed by a modifier instead of a procedurename in which case it acts accordingly.

PO ALL (command)

Prints the text of all (unburied) procedures, all names, and the specifications of all arrays in that order.

PO ARRAY *arrayname* (command)

Prints the named array's size and type.

PO ARRAYS (command)

Prints the size and type of all currently defined arrays.

PO FILE *filename* (file command)
POF *filename*

Prints the entire contents of the file.

PO INDEX (file command)
POI

Prints the names of all files and subindices in the local index.

PO LINE *linenumber* (editing command)
POL *linenumber*

Prints out the specified line.

PO NAMES (command)

Prints out all currently defined names and their values. To print the value of one particular name, use PR.

PO PROCEDURES (command)

Prints the text of all (unburied) procedures in the workspace.

PO TITLE (editing command)
POT

Prints the title of the procedure currently being edited or defined.

POTS (command)

Prints the titles of all (unburied) procedures in the workspace line by line.

PO TREE (file command)

Prints the tree of the current index. (See TREE.)

PROCEDURES (modifier)

Used with PO and ERASE to limit the commands to procedures only.

PRODUCT *1stfactor 2ndfactor* (operation)

Outputs the PRODUCT of *firstfactor* and *secondfactor*. See section on INFIX OPERATORS in the appendix.

.PWRCLR (command)

Used in the hopefully rare occasions when the TTY controller or display controller screws up. You might try this if your console allows you to type CS or FD 100, but does not display a turtle. More likely, however, your display is simply switched to a different line. Call for help then. Also useful if all TTYs except the main console seem to "freeze up" while LOGO is apparently still running. This command resets the speed of all TTY lines to their normal speed and may screw some user (especially one using a Datapoint console), so be careful.

QUOTIENT *dividend divisor* (operation)

Outputs *dividend/divisor*. If both inputs are integers, this outputs the integer part of the quotient. See section on INFIX OPERATORS in the appendix.

RANDOM (operation)

Outputs a one-digit random integer.

READ *filename* (file command)

Reads the specified file into the workspace. See FILE for more details.

READPTR (command)

Reads into the workspace from the paper tape reader.

RELAY *bluebox 0or1* (command)

Sends *0or1* to the relay box numbered *bluebox*.

REMAINDER *dividend divisor* (operation)

Outputs *dividend* (MOD *divisor*). See section on INFIX OPERATORS in the appendix.

REQUEST (operation)

When executed, this operation waits for a line to be typed in from the console. It then outputs that line as a list. See also TYPEIN. The prompt character for a request is < if nothing else was typed at the beginning of the current line, but is omitted altogether if the request is begun in the middle of a line.

```
?TO TAKENOTES
>10 .OPENA "NOTEBOOK
>20 TYPE "TITLE:%%
>30 MAKE "A REQUEST
>40 .FILEP SE DATE :A
>50 MAKE "A REQUEST
>60 IF REQUEST = [.] .FILEP [***] .CLOSEF STOP
>70 .FILEP :A
>80 GO 50
>END
TAKENOTES DEFINED
?TAKENOTES
TITLE:  MEMORANDUM
<THIS IS A WAY
<TO LEAVE BUG-MAIL
<AND OTHER NOTES TO YOURSELF
<.
?POF "NOTEBOOK
75 1 11 RANDOM MEMO
PREVIOUS MESSAGES...
***
75 1 17 MEMORANDUM
THIS IS A WAY
TO LEAVE BUG-MAIL
AND OTHER NOTES TO YOURSELF
***
?
```

RIGHT *degrees* (turtle command)

Causes the floor or display turtle to rotate *degrees* to the right.

RTOUCH (TS-turtle operation)

Outputs "TRUE if the right of the touch-sensor turtle is touching something. Outputs "FALSE otherwise. Like FTOUCH, LTOUCH, and BTOUCH.

.RUG (status command)

Stops LOGO by breaking out to RUG. This stops EVERYbody, so don't use it unless you know what you are doing.

RUN *list* (command or operation)

Evaluates the input as if it had been typed at the console. RUN outputs when it evaluates an operation and does not when it evaluates a command, but note the following special situations (and see EDIT for more clarification).

```
?TO UNTIL :CONDITION :ACTION
>10 IF RUN :CONDITION STOP
>20 RUN :ACTION
>30 GO 10
>END
UNTIL DEFINED
?UNTIL [FTOUCH] [FD 10]    ! moves TS-turtle forward until it touches something
```

RUN is also necessary for the execution of REQUESTs and the execution of computed TO, ED, ER, PO, BURY, STEP or TRACE, that is, the commands which quote their inputs. (See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.) None of these will do the right thing if its input is a name or a procedure which outputs a name. For example:

```
?ER :A
:A - ERASE WHAT??
?BURY :A ! BURY also takes its input literally.
PROCEDURE :A NOT HERE.
?RUN SE "BURY :A ! But this works!
?
```

SEND *usernum message* (command)

Prints *message* (word or list) on console *usernum* (or waits until the user of that console is not typing before sending the message.)

SENTENCE *thing1 thing2* (operation)

If both inputs are lists, it strings the elements of the two together to make a single list and outputs that. If either input is not a list, it first changes that input to a one-element list and then proceeds as above. This operation may accept other than two inputs if surrounded by parentheses. Compare the examples below with FPUT and LIST respectively.

```
?FPRINT ( SE "WORD [AND LIST] "ANDWORD [WITH A LIST] )
[WORD AND LIST ANDWORD WITH A LIST]
?FPRINT ( SE ( SE [A B] "C [D E] ) )
[A B C D E]
?
```

SETASIZE *nodes* (command)

Used for grabbing *more* arrayspace when needed, this command reinitializes your workspace first (almost like saying GOODBYE) and is thus to be used with thought. To compute *nodes* properly, count 16 nodes for every array and 1 node for every entry in a pointer array, 2 nodes for every entry in a floating or integer array. It is not necessary (and useless) to use this unless the normal allocation of array space is insufficient. If you give yourself too much, you begin to lose regular node space, so beware.

SETHEADING *degrees* (display command)
Sets the heading of the display or plotter turtle to *degrees*.

SETINDEX *indexname* (file command)
SETI *indexname*
Sets the default index to *indexname*. See INDEX and FILE for more details.

SETTURTLE *statelist* (display command)
SETT *statelist*
The list statelist must contain three elements in the format [*x-coord y-coord heading*] and the command sets the display or plotter turtle's whereabouts accordingly.

.SETTV *whichvar value* (command)
Sets the Time and date Variables for the TIME and DATE operations. *Whichvar* is a number from 0 through 5 and specifies seconds through years in that fashion. *Value* is the value you want that variable set to.

SETX *x-coord* (display command)
Moves the display or plotter turtle horizontally to the specified coordinate.

SETXY *x-coord y-coord* (display command)
Moves the display or plotter turtle to the specified coordinates.

SETY *y-coord* (display command)
Moves the display or plotter turtle vertically to the specified coordinate.

.SGCF (status switch)
Set Garbage Collect Flag so that all litter is removed the very instant it is created. This slows you down beyond belief and has little other value except for system debugging. Cleared by .CGCF

SHOWTURTLE (display command)
ST
Causes the display turtle to appear.

SIN *degrees* (operation)
Outputs the sine of *degrees*.

SNAP (display operation)

Used in expression MAKE "SOMENAME SNAP to assign the specified name to the current contents of the display screen (or to the contents since the last NEWSNAP). Once the name has been assigned, DISPLAY :SOMENAME can be used to cause another copy of the picture to appear, its location being offset from the original by the amount that the turtle is currently offset from the origin (usually (0,0), but see NEWSNAP). Similarly, WIPE :SOMENAME will remove all copies of :SOMENAME from the display without otherwise affecting the contents of the screen. SNAPs cannot be written into files.

.SPNF (switch)

Set Print Nodes Flag so that you will be told how many nodes you have immediately after each spontaneous or evoked garbage collection. Cleared by .CPNF

SQRT *number* (operation)

Outputs the square root of the number.

STARTDISPLAY *0or1* (switch)

Grabs and initializes a display. STARTDISPLAY 0 is essentially the same as CS PD ST and STARTDISPLAY 1 also allocates twice the usual display space (allows more lines on the screen and more snaps to be created). Existing snaps do not survive through STARTDISPLAY and cause an error if used afterward.

.STATUS (status switch)

A binary switch that unprotects all status-protected words.

STEP *procedurename* (switch)

Allows the user to monitor the execution of the specified procedure (or all procedures if *procedurename* is ALL) by 1) printing at the user's console the inputs to the procedure and its outputs if any; 2) allowing the user to approve, reject or postpone each line of the procedure by responding with CARriage RETurn, CTRL G or CTRL Z after the machine types the line. Carriage return executes the line, CTRL G stops the execution entirely (but does not unSTEP the procedure for subsequent executions) and CTRL Z holds everything in its current state and allows the user to interpose other commands until he wishes to continue (see CONTINUE). To unSTEP, use ER STEP *procedurename*. STEP quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.

.STF (switch)

Sets Trace Flag to show cruft that is incomprehensible to all but the elite. Cleared by .CTF

STOP (flow)

Inside a procedure it stops the procedure and returns control to the next higher level. See also PAUSE and TOPLEVEL.

STORE *arrayname coords item* (command)

As in DEFAR and GET, if the array is more than one-dimensional, STORE and its inputs must be enclosed in parentheses. This stores *item* at *coords* in *arrayname*.

SUM *addend1 addend2* (operation)

Outputs the sum of its inputs. It can accept a variable number of inputs if it is parenthesized.

SWITCH *boxnum switchnum* (operation)

Outputs "TRUE if switch number *switchnum* on box number *boxnum* is on and "FALSE otherwise. See appendix for a diagram of the switchbox switch numbers.

SYSPR *message* (command)

Tries to print *message* on everybody's console, waiting for individuals to finish typing at their console if necessary.

TBOX (switch)

Declares to the computer that you are a Thornton Box. The computer responds by asking you, one port at a time, what devices are connected to what TBox ports. You respond with the letters N, M, T, C, or P, which mean, respectively Null (no device), Music, Turtle, Console, and Plotter. If you don't claim to have a console on any port the computer will know you are lying and will complain.

TEST *TorF* (command)

TEST runs its input, which must evaluate to "TRUE or "FALSE and saves the result in a special place, local to the procedure in which the test was made, which can be read and used by IFTRUE and IFFALSE. The results remain unchanged until a subsequent TEST or until the locale ceases to exist.

TEXT *procedurename* (operation)

Outputs a list containing the text of the named procedure. Each element of the list is a list containing one line of the procedure, beginning with the title and not including the line [END]. The following procedures illustrate a way of using TEXT to file procedures in their proper place. (See example next page.)

(continued from last page, example using TEXT)

```
?TO FILETHEM :PLIST ! AIDS                          ! the comment is a "tag"
>10 IF EMPTYP :PLIST PR "***DONE*** STOP
>20 MAKE "TEMP TEXT F :PLIST                        ! grabs text
>30 MAKE "TOLINE F :TEMP                            ! grabs title line
>40 IF MEMBERP "! :TOLINE                           ! is file labeled?
        MAKE "PNAME F BF :TOLINE                    ! procedurename
        MAKE "FILEN L :TOLINE                       ! file name
      ELSE GO 100                          ! if not labeled, skip it
>50 PUTAWAY :PNAME :FILEN
>100 PR SE :PNAME [NOT FILED]
>110 FILETHEM BF :PLIST
>END
FILETHEM DEFINED
?TO PUTAWAY :PNAME :FILEN ! AIDS
>10 .OPENA :FILEN
>20 IF EMPTYP :TEMP GO 100                          ! remember [END]
>30 .FILEP F :TEMP
>40 MAKE "TEMP BF :TEMP
>50 GO 20
>100 .FILEP [END]
>110 .CLOSEF                                        ! notify user
>120 PR ( SE :PNAME [FILED IN] WORD "" :FILEN )
>END
PUTAWAY DEFINED
?POTS
TO MEMBERP :ITEM :LIST
TO FILETHEM :PLIST ! AIDS
TO PLAYSCALE ! MUSIC
TO PUTAWAY :PNAME :FILEN ! AIDS
TO SHOWSCALE ! SCREENMUS
?FILETHEM CONTENTS
MEMBERP NOT FILED
FILETHEM FILED IN "AIDS
PLAYSCALE FILED IN "MUSIC
PUTAWAY FILED IN "AIDS
SHOWSCALE FILED IN "SCREENMUS
***DONE***
?
```

THEN (noise word)

Unnecessary but occasionally attractive separator between the conditional clause of an IF-line and the statement to be executed conditionally. See IF.

THING *name* (operation)

Outputs the thing associated with *name*, the value assigned to *name*.

THINGP *name* (operation)

Outputs "TRUE if there currently exists some thing with the name *name* and outputs "FALSE otherwise. Example:

```
?TO HI
>10 IF THINGP "USERNAME GO 100                    ! do I need to ask?
>20 TYPE [WHAT IS YOUR NAME?%%%]     ! if so, ask.
>30 MAKE "USERNAME REQUEST
>40 PRINT "HI! STOP
>100  PR SE "HI, WORD :USERNAME "!     ! else just greet her
>END
?HI DEFINED
?HI
WHAT IS YOUR NAME?  BOBBIE
HI!
?HI                                    ! this time, it already knows the name
HI, BOBBIE!
?
```

TIME (operation)

Outputs a three-element list which might contain the correct time [hr min sec] but probably will not. Once set correctly (using .SETTV and a 24 hour clock), TIME will tell the truth as long as the system stays up.

TO *title* (switch)

Used to define procedures. The input, *title*, must contain a name not already used by the user or by LOGO and may contain an indefinite number of additional words prefixed by : which name inputs to the new procedure. Turns on editing mode (see also EDIT) and is terminated by the switch END. (See throughout, but especially TEXT for examples.) TO quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.

TOOT *twicenum* (command)

Tells the floor turtle to toot its horn *twicenum*/2 times.

TOPLEVEL (flow)

Stops the procedure which invoked it and returns control to TOP level.

TRACE *procedurename* (switch)

Causes the computer to type a message each time the named procedure is executed and indicate what if any inputs the procedure received and what it outputs when it stops. TRACE ALL traces all procedures. ER TRACE *procedurename* stops tracing that procedure. (And, of course, ER TRACE ALL gets rid of all traces.) TRACE quotes its input. See PRIMITIVES WHICH QUOTE THEIR INPUTS in the appendix.

TREE (tree)

Used only in the context PO TREE (which see under PRINTOUT). The filing system begins with a "root directory" on each disk which simply names the disk. The fixed platter of the moving-head disk is called "FIXED; (don't forget the ;) and the removable platter of the disk is called "A101; (never mind why). The "root" directories are the ones you get when you say SETI "FIXED; or SETI "A101; or HELLO (which puts you on the root of the disk called "FIXED;). These directories contain some files and some sub-indices. For example, the index "FIXED; contains the "NEWS file that prints out messages when you say HELLO to LOGO. Both roots contain an index called USERS which contain indices of users which may contain files and sub-indices which may contain other files and sub-indices which may.... This branched structure is called a TREE. Whereas, the command PO INDEX prints out the current index and shows what sub-indices it may have, it does not show the *contents* of these sub-indices. On the other hand, PO TREE starts at the current index and prints the complete structure of the tree from there down. (Note that the "root" is, unlike ordinary trees, at the top, and that the up-arrow sign, ", is used for moving up to a higher index. See section on LOGO SPECIAL NAMES in the appendix.) For further clarification of the filing structure, see FILE and the specific filing commands.

TTYP (operation)

Outputs "TRUE if anything (including space or delete) was typed at the user's keyboard since the last REQUEST, TYPEIN or .CTYI. (Those operations grab what was typed in and *clear out* the teletype buffer. Note that at present there is no way to know *what* was typed in without using one of those operations and that TTYP does not tell you if anything *now* was typed since the last time you looked in the buffer.) Outputs "FALSE otherwise.

```
?TO LIMIT :SECONDS                          ! limits typing time
>10 MAKE "TIME TIME
>20 MAKE "ANSWER TIMEREQUEST :SECONDS
>30 IF NOT ( CLOCK - :PREVIOUS ) / 60 > :SECONDS OP :ANSWER
>40 PRINT [TOO MUCH TIME!]
>50 OP []
>END
LIMIT DEFINED
?TO TIMEREQUEST :SECONDS                    ! limits response time
>10 IF :SECONDS < 1 OUTPUT []
>20 MAKE "PREVIOUS LAST TIME
>30 IF TTYP MAKE "PREVIOUS CLOCK OUTPUT REQUEST
>40 IF LAST TIME = :PREVIOUS GO 10
>50 TIMEREQUEST :SECONDS - 1
>END
TIMEREQUEST DEFINED
?
```

Once we commit ourselves to REQUEST or TYPEIN, we are stuck until a carriage-return is typed and no further interaction with the clock is possible. With .CTYI, every character is cleared from the buffer before another is checked by TTYP and so constant interaction with the procedure becomes possible, as in the following example. (see example on next page)

(continued from last page)

```
?TO ORBIT
>10 INIT
>20 MOVESATELLITE
>30 IF TTYP MAKE "A .CTYI
>40 IF :A = 7 STOP
>50 IF MEMBERP CHAR :A [J D F B] THRUST :A
>60 GO 20
>END
ORBIT DEFINED
?
```

TURTLE *1or2* (switch)

Specifies turtle 1 or 2 as the device to which all subsequent turtle commands will be sent. Turned off by requesting a different turtle or the plotter or display or by saying NOTURTLE.

.TYI *device* (operation)

Like .CTYI but with the device or teletype specified by *device*. See .CLOSEfor a list of device names this understands.

.TYO *device ascii* (command)

Like .CTYO but outputting the character whose decimal ascii code is *ascii* to device *device*. The way the device handles the character depends on the device itself. See index for a table of characters the floorturtle understands. To free the device for others to use it, you must do a .CLOSE (which you should see for a list of special device names this command understands).

TYOWAIT *device extrawait* (flow)

Causes your program to wait *extrawait* 30ths of a second after the specified device is finished doing what it was told to do. This recognizes the same *devices* that .CLOSE understands.

TYPE *anything* (command)

Like PRINT except does not carriage return at the end of the line.

TYPEIN (operation)

Like REQUEST, this operation waits for a line to be typed in at the console, but regardless of the nature of the typed line, this outputs only the first word as a word. TYPEIN outputs the empty word if only a carriage return is typed.

USE *username* (file command)

Establishes the index that will apply to subsequent file commands such as READ, WRITE, POF and POI. If the input is a single name (the usual case), that name is interpreted as a user on the current disk. If the input is a list, the first element is the user name and the rest is a legal tree of his index. The last element of the list is the default index. See FILE for more details.

.USERTIME (operation)

Outputs a number which is just about totally useless. It is, for the interested, the total time used by all users since the system came up. The units are 60ths of a second, maybe.

.VALUE (status operation)

Even the elite find this of little value. This gives you the value of the top thing on the S-PDL.

.VERSION (operation)

Outputs the current version number of LOGO.

VLEN (music operation)

Outputs the total length of music compiled for the current voice. See NOTE for more details.

VOICE *voicenum* (switch)

Specifies the voice to which all subsequent NOTE commands apply. If no VOICE command is given, voice 0 is assumed. Specification persists until changed.

WAIT *num* (command)

Causes your program to wait *num* 60ths of a second before continuing.

WIPE *snapname* (display command)

Erases all appearances of the snap from the display screen.

WIPECLEAN (display command)

Erases everything from the screen, but leaves the turtle in its current state.

WORD *word1 word2* (operation)

Outputs a word composed of the two inputs in order. With parentheses, a variable number of inputs (see appendix) may be used. Any input may be a number instead of a word.

WORDP *thing* (operation)

Outputs "TRUE if *thing* is a word and "FALSE otherwise.

WRAP (display switch)

Allows the display turtle (and images) to "wrap around" the screen to the other side. (Gives you a toroidal screen.)

WRITE *filename* (file command)

Creates a file with the name *filename* in the current index. The file will contain all names and all unburied procedures in your workspace at the time. Neither arrays nor snaps can be written into a file.

WRITEPTP (command)

Punches a paper tape of the contents of your workspace (except snaps and arrays). Occasionally useful for permanent storage of a file you want to delete from disk. Make sure PaperTape Punch is ready before executing.

XCOR (display operation)

Outputs the x-coordinate of the display turtle.

YCOR (display operation)

Outputs the y-coordinate of the display turtle.

VARIABLE NUMBER OF INPUTS

Certain primitives can take a variable number of inputs. To use this feature, the *primitive and its inputs* must be grouped together in parentheses. For example,

(SUM 5 6 7 8) outputs 26

The primitives having this feature are:

DEFINEARRAY	STORE	GET
PRINT	TYPE	FPRINT
LIST	WORD	SENTENCE
PUT	LPUT	
NOTE	MUTYO	
SUM	PRODUCT	

PRIMITIVES WHICH QUOTE THEIR INPUTS

Certain LOGO primitives cause the following word not to be evaluated regardless of the nature of that word. E.g., to bury or erase several items on a list, it is not possible to say BURY FIRST :PROCEDURELIST because BURY will see the word FIRST, assume it is the name of a procedure, attempt to bury it, and cause an error before FIRST has a chance te get the real procedure out of the list. Yet these primitives will accept quoted inputs; BURY ORBIT and BURY "ORBIT are equivalent. This enables computed buries, erases, edits, etc. All must be executed in the form:

?RUN SE [ER BURY] FIRST :PROCEDURELIST

The primitives having this feature are:

BURY	ERASE BURY
STEP	ERASE STEP
TRACE	ERASE TRACE
TO	EDIT
PRINTOUT	

INFIX OPERATORS

* infix PRODUCT
\+ infix SUM
\- infix DIFFERENCE
/ infix QUOTIENT
> infix GREATER
< infix LESS
= infix EQUAL
\ infix MOD
infix MAKE, ie., "A 5 is equivalent to MAKE "A 5

EDITING CHARACTERS (not restricted to edit mode)

<rubout> Deletes the previous character.
^W Deletes the previous word.
^Y Places the previous line typed in the edit buffer

If something is in the "edit buffer" (that is, after ^Y or EDL), the following may be used:

^C Copy the next character.
^N Copy the Next word.
^R Copy the Rest of the edit buffer.
^D Delete the next character.
^S Skip the next word.
<carriage return> Terminates current line as it appears on console.

SPECIAL CHARACTERS

% PRINTs as a space. (But PRINTOUT shows it as a %).
! Used for comments. Anything appearing between it and next ! on a line is ignored.
Takes one input which must be a word and runs it as a procedure, e.g.,

```
# WORD "POOH 5
```

will execute the procedure named POOH5.

: Same as THING.
( ) Used for grouping.
[ ] Used to indicate lists.
^A Echoes as carriage return.
^B Echoes as blank.
^G Break. Interrupts procedure and acts immediately when typed. (But see .CTYI)
^H Echoes as backspace.
^I Echoes as tab.
^J Echoes as linefeed.
^M Echoes as carriage return.
^Q Superquotes the following character (i.e., removes its special features).
^X Clarifies line currently in edit or typein buffer.
^Z Causes a pause. Interrupt status, like ^G.

LOGO SPECIAL NAMES

"^ Refers to index above current index. (Can be used with SETI.)
"; Refers to root index of current disk. (Can be used with SETI.)
"A101; Refers to root of moving-head disk.
"FIXED; Refers to root of fixed-head disk.
"TRUE Output of predicates that evaluate to True. Accepted by IF.
"FALSE Output of predicates that evaluate to False. Accepted by IF.
"MUSIC Device name for .TYO, .CLOSE, TYOWAIT.
"TUR1 Device name for .TYO, .CLOSE, TYOWAIT.
"TUR2 Device name for .TYO, .CLOSE, TYOWAIT.
"PLOTTER Device name for .TYO, .CLOSE, TYOWAIT.
"TTY Device name for .TYO, .CLOSE, TYOWAIT.

DECIMAL ASCII CODE AND CORRESPONDING CHARACTERS

0	(^@)	32	*space*	64	@	96	`
1	(^A)	33	!	65	A	97	a
2	^B	34	"	66	B	98	b
3	^C	35	#	67	C	99	c
4	^D	36	$	68	D	100	d
5	^E	37	(%)	69	E	101	e
6	^F	38	&	70	F	102	f
7	^G *bell*	39	'	71	G	103	g
8	*backspace*	40	(	72	H	104	h
9	*tab*	41	)	73	I	105	i
10	*linefeed*	42	*	74	J	106	j
11	(^K)	43	+	75	K	107	k
12	(^L)	44	,	76	L	108	l
13	*return*	45	-	77	M	109	m
14	^N	46	.	78	N	110	n
15	^O	47	/	79	O	111	o
16	^P	48	0	80	P	112	p
17	^Q	49	1	81	Q	113	q
18	^R	50	2	82	R	114	r
19	^S	51	3	83	S	115	s
20	^T	52	4	84	T	116	t
21	^U	53	5	85	U	117	u
22	^V	54	6	86	V	118	v
23	^W	55	7	87	W	119	w
24	^X	56	8	88	X	120	x
25	^Y	57	9	89	Y	121	y
26	^Z	58	:	90	Z	122	z
27	^[	59	;	91	[	123	{
28	^\	60	<	92	\	124	|
29	^]	61	=	93	]	125	}
30	^^	62	>	94	^	126	~
31	^_	63	?	95	_	127	*(delete)*

Ascii 0, 1, 11, 12 and 127 do not type on the console even with TYPE CHAR 12 and some others do not echo normally. The italicized words above indicate special echoing or special functions. See section on special characters.